FINAL REPORT FOR

"Rapidly Customizable Spoken Dialogue Systems"

(N00014-05-1-0314)

James Allen

Florida Institute for Human and Machine Cognition (IHMC)

Submitted to:

Dr. Behzad Kamgar-Parsi
Office of Naval Research
Code 311
875 N. Randolph St.
Arlington VA 22203-1995

ihmc
INSTITUTE FOR HUMAN & MACHINE COGNITION
40 South Alcaniz Street, Pensacola, FL 32502

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)*
28-01-2009

2. REPORT TYPE
Final Technical Report

3. DATES COVERED *(From - To)*
01-06-2005 to 31-05-2008

4. TITLE AND SUBTITLE
Rapidly Customizable Spoken Dialogue Systems

5a. CONTRACT NUMBER

5b. GRANT NUMBER
N00014-05-1-0314

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Dr. James Allen

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Florida Institute for Human and Machine Cognition
40 S. Alcaniz St.
Pensacola FL 32502

8. PERFORMING ORGANIZATION REPORT NUMBER
Final

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
One Liberty Center
875 N. Randolph St.
Arlington VA 22203-1995

10. SPONSOR/MONITOR'S ACRONYM(S)
ONR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unrestricted

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Building a robust spoken dialogue system for a new application, task, or domain currently requires considerable effort, including substantial efforts in data collection, building language models, grammar/parser development, building a custom dialogue manager, and developing the connection to the system's "back-end" systems (e.g., a database query or knowledge based system). This project developed key parts of a technology base upon which spoken dialogue systems can be rapidly constructed for new domains. Our approach involves building generic components (i.e., ones that apply in any practical domain) for all stages of spoken dialogue understanding, and developing techniques for rapidly customizing the generic components to new domains. To achieve this goal we made progress in several important areas: (1) developing a generic domain-independent grammar of spoken English together with techniques for optimizing parser performance for specific domains, (2) a domain independent representation of semantic meaning with an ontology mapping framework that allows the user to define relatively simple mapping rules to the domain-specific communication/representation language, and (3) a domain-general collaborative problem solving framework that enables rapid construction of the dialogue agents, and provides the link to domain-specific reasoning capabilities.

15. SUBJECT TERMS
Spoken Dialogues, Artificial Intelligence, Human-Centered Computing, Language Models, Domain-independent grammar

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			James Allen
U	U	U	UU		19b. TELEPHONE NUMBER *(Include area code)* 850-202-4400

INSTRUCTIONS FOR COMPLETING SF 298

1. REPORT DATE. Full publication date, including day, month, if available. Must cite at least the year and be Year 2000 compliant, e.g. 30-06-1998; xx-06-1998; xx-xx-1998.

2. REPORT TYPE. State the type of report, such as final, technical, interim, memorandum, master's thesis, progress, quarterly, research, special, group study, etc.

3. DATES COVERED. Indicate the time during which the work was performed and the report was written, e.g., Jun 1997 - Jun 1998; 1-10 Jun 1996; May - Nov 1998; Nov 1998.

4. TITLE. Enter title and subtitle with volume number and part number, if applicable. On classified documents, enter the title classification in parentheses.

5a. CONTRACT NUMBER. Enter all contract numbers as they appear in the report, e.g. F33615-86-C-5169.

5b. GRANT NUMBER. Enter all grant numbers as they appear in the report, e.g. AFOSR-82-1234.

5c. PROGRAM ELEMENT NUMBER. Enter all program element numbers as they appear in the report, e.g. 61101A.

5d. PROJECT NUMBER. Enter all project numbers as they appear in the report, e.g. 1F665702D1257; ILIR.

5e. TASK NUMBER. Enter all task numbers as they appear in the report, e.g. 05; RF0330201; T4112.

5f. WORK UNIT NUMBER. Enter all work unit numbers as they appear in the report, e.g. 001; AFAPL30480105.

6. AUTHOR(S). Enter name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. The form of entry is the last name, first name, middle initial, and additional qualifiers separated by commas, e.g. Smith, Richard, J, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES). Self-explanatory.

8. PERFORMING ORGANIZATION REPORT NUMBER. Enter all unique alphanumeric report numbers assigned by the performing organization, e.g. BRL-1234; AFWL-TR-85-4017-Vol-21-PT-2.

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES). Enter the name and address of the organization(s) financially responsible for and monitoring the work.

10. SPONSOR/MONITOR'S ACRONYM(S). Enter, if available, e.g. BRL, ARDEC, NADC.

11. SPONSOR/MONITOR'S REPORT NUMBER(S). Enter report number as assigned by the sponsoring/ monitoring agency, if available, e.g. BRL-TR-829; -215.

12. DISTRIBUTION/AVAILABILITY STATEMENT. Use agency-mandated availability statements to indicate the public availability or distribution limitations of the report. If additional limitations/ restrictions or special markings are indicated, follow agency authorization procedures, e.g. RD/FRD, PROPIN, ITAR, etc. Include copyright information.

13. SUPPLEMENTARY NOTES. Enter information not included elsewhere such as: prepared in cooperation with; translation of; report supersedes; old edition number, etc.

14. ABSTRACT. A brief (approximately 200 words) factual summary of the most significant information.

15. SUBJECT TERMS. Key words or phrases identifying major concepts in the report.

16. SECURITY CLASSIFICATION. Enter security classification in accordance with security classification regulations, e.g. U, C, S, etc. If this form contains classified information, stamp classification level on the top and bottom of this page.

17. LIMITATION OF ABSTRACT. This block must be completed to assign a distribution limitation to the abstract. Enter UU (Unclassified Unlimited) or SAR (Same as Report). An entry in this block is necessary if the abstract is to be limited.

Final Report

Rapidly Customizable Spoken Dialogue Systems

Florida Institute for Human and Machine Cognition

1. Executive Summary

Building a robust spoken dialogue system for a new application, task, or domain currently requires considerable effort, including substantial efforts in data collection, building language models, grammar/parser development, building a custom dialogue manager, and developing the connection to the system's "back-end" systems (e.g., a database query or knowledge based system). This project developed key parts of a technology base upon which spoken dialogue systems can be rapidly constructed for new domains. Our approach involves building *generic* components (i.e., ones that apply in any practical domain) for all stages of spoken dialogue understanding, and developing techniques for rapidly customizing the generic components to new domains. To achieve this goal we made progress in several important areas: (1) developing a generic domain-independent grammar of spoken English together with techniques for optimizing parser performance for specific domains, (2) a domain independent representation of semantic meaning with an ontology mapping framework that allows the user to define relatively simple mapping rules to the domain-specific communication/representation language, and (3) a domain-general collaborative problem solving framework that enables rapid construction of the dialogue agents, and provides the link to domain-specific reasoning capabilities.

During this project, we used the generic technology developed to enable the construction of a dialogue-based task learning system called PLOW. A paper based on this system won the outstanding paper award at the annual conference of the Association for the Advancement of Artificial Intelligence (AAAI) in 2007 (Allen et al, 2007). A core component of that system is a domain-general deep language understanding system. A key accomplishment in this effort was developing techniques to enable broad-coverage deep understanding by taking advantage of many recent developments in statistical techniques and corpora. Typically used only for shallow understanding. Our preliminary experiments on parsing previously unseen text indicates great promise for the work (Allen et al, 2008).

In the remainder of this report, we describe these accomplishments in more detail.

2. Broad-Coverage Deep Natural Language Understanding

Deep language understanding involves mapping language to expressions capturing its intended meaning, in terms of concepts and relations in an ontology that supports reasoning. Deep understanding is needed in many applications, including dialogue-based human-computer interfaces to intelligent systems/agents, tutoring and advice-giving systems, systems that learn from instruction, and systems that learn from reading.

There seems to be a consensus in the field that broad-coverage, high-accuracy deep parsing is currently not feasible. We do not believe this is the case and discuss here the core generic tech-

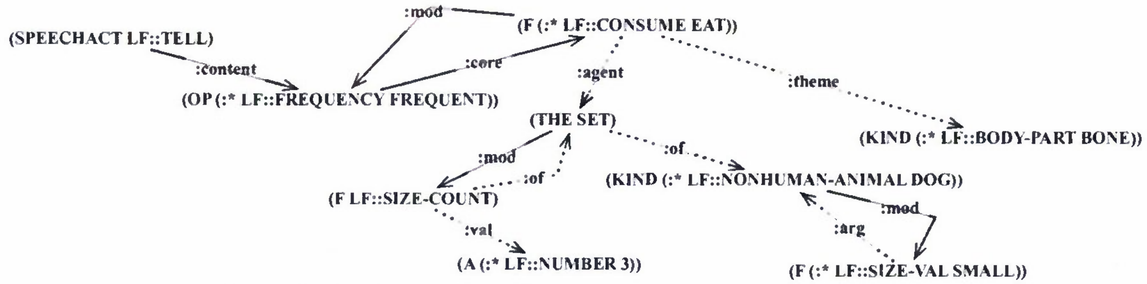


Figure 1: The LF graph for "The three small dogs frequently eat bones"

nologies we developed for deep semantic processing of natural language. In order to attain high-accuracy broad-coverage deep processing, we augmented the core system with statistical processing to aid in disambiguation, and large-scale lexical resources to extend the lexicon. In this way, the deep understanding system can be guided by a wide range of advice derived from statistical language processing, including named-entity recognizers, statistical parsers, word sense disambiguation techniques and semantic role identification, plus a large base of shallow generic knowledge. In other words, the deep parser provides the framework to integrate all the results from a diverse range of statistical models into a consistent deep logical form. Initial experiments suggest that this approach has great promise.

The Logical Form

The logical form (LF) is the semantic representation language produced by the parser. It is designed to be an expressive, yet intuitive, formalism for expressing sentence logical form. In designing the LF, we had multiple considerations: (1) it needs to be expressive, providing good coverage of the complex semantic phenomena in language, including modal operators, generalized quantifiers, and underspecified scoping constraints (cf MRS (Copestake et al., 2005)); (2) it needs to be fully indexed into the word senses in the semantic ontology, as opposed to using uninterpreted predicates found in many logical forms; (3) it needs to support robust processing of sentence fragments that are common in speech; (4) it needs to support the implementation of ontology-mapping rules; and (5) it needs to be understandable to humans - readability of formalisms is critical for debugging and analysis.

We define the LF in its graphical form. Besides being more intuitive, the graphical form allows interesting comparisons to approaches that produce partial semantic analyses, such as statistical word-sense and semantic role disambiguation techniques. In addition, the graphical formalism leads to easier formal analysis. Consider the LF for the sentence *The three small dogs frequently eat bones* shown in Figure 1. There are many types of objects evoked by this sentence, captured by the nodes in the graph. First, there is the event of the dogs liking bones, where the node captures a reified event in a Davidsonian-style (Davidson, 1967) representation. Next we have properties like small, which are reified in the same way. The interpretation of the three small dogs requires several nodes, including a set of size three, consisting of dogs that are small (rather than the set being small). Furthermore, as a definite description, we expect to be able to identify the set of dogs from the discourse context. Finally, we need to capture that bones refers to a kind of object rather than, say, a specific set of bones. Note that each node indicates a specifier (indicating the type of node, be it a generalized quantifier, event identifier, or kind) as well as the type of the object. This is critical for subsequent discourse processing. Nodes are connected by arcs that indicate argument relations (semantic roles from the LF ontology) and dependency relationships (critical for resolving the unscoped LF into a fully scoped formal representation). There are two types of arcs: those connecting to terms and those connecting to predicate/formulas. The distinction between them is important for the quantifier scoping algorithm.

The LF formalism has additional features to capture aspects such as coreference relations, implicit arguments to predicates, complex quantification (e.g., *almost all dogs*, *every other dog*, *all but one*), modals, tense, aspect, negation, complex adverbials, numbers, time and date expressions, and other complicated phenomena.

The Core Parsing Technology: The grammar is a lexicalized context-free grammar, augmented with feature structures and feature unification. The grammar is motivated from X-bar theory, and draws on principles from GPSG (e.g., head and foot features) and HPSG. While it has a context-free backbone, the parsing is best seen as a search through possible logical forms. The search in the parser is pruned by domain-general selectional restrictions from the ontology to eliminate semantically anomalous sense combinations during parsing. The parser builds constituent/logical forms bottom-up using a best-first search strategy similar to A*, combining pre-specified rule and lexical weights and the influences of the statistical techniques described below. The search terminates when a pre-specified number of spanning constituents have been found or a pre-specified maximum chart size is reached. The chart is then searched using a dynamic programming algorithm to find the least cost sequence of constituent/logical forms according to a scoring function that can be varied by genre being processed.

The current lexicon contains approximately 7,000 hand built lexical lemmas (with morphological variants, yielding 17000 words), each identified with a semantic concept in the LF ontology that specifies the selectional restrictions on its possible arguments and modifiers.

The Broad Coverage system: To attain broader coverage, we used input from a variety of external resources. We built a subsystem for unknown word lookup that accesses lexical resources such as Wordnet (Miller, 1995) and Comlex (Macleod et al., 1994). The WordNet senses are mapped to the LF ontology at an abstract level and the combined resource information is used to build lexical entries with approximate semantic and syntactic structures for words not in the core lexicon. Because the information in such entries is underspecified, the parser must deal with significantly increased levels of ambiguity when dealing with such words.

Because it was developed for speech applications, the parser is designed to accept word lattices as input so speech recognizers can pre-populate the chart with different word hypotheses, letting the parser choose among them based on what entries make the best overall interpretations. We use the same mechanism for integrating a corpora-based preprocessors such as a named entity recognizer, which adds hypotheses to the input chart about possible named entities. Note these are hypotheses–the parser does not have to use them. As with word hypotheses from a speech recognizer, the parser will choose the input hypotheses that lead to the best overall interpretation. In addition, we can use statistical part-of-speech and word sense disambiguation techniques to

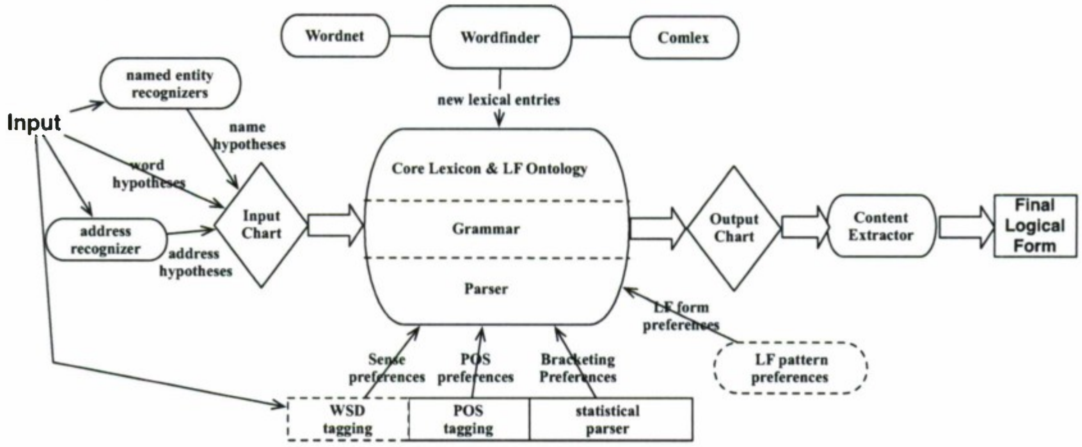


Figure 2: Extending and Guiding Deep Parsing

suggest likely interpretations of words in the input chart. Using techniques similar to Swift et al. (2004) and Cahill et al. (2007), the extended system also receives constituent structure advice from a state-of-the-art statistical parser. For the results reported here, we used the out-of-the-box unlexicalized stochastic context-free grammar parser from Stanford (Klein and Manning, 2003). Again, these are preferences that help guide parsing, but do not limit the range of possible overall interpretations. The system with these extensions is shown in Figure 2. The parts with dotted outlines are under development and not included in the current evaluations

Evaluation

We performed an evaluation of the coverage and accuracy of the extended parser on seven paragraphs (Text 1-7) submitted by seven different research groups for a common evaluation at the workshop on the semantics of text processing (Bos and Delmonte, 2008). Below is a sample paragraph, Text #6, which proved the most challenging for our system:

> *Amid the tightly packed row houses of North Philadelphia, a pioneering urban farm is providing fresh local food for a community that often lacks it, and making money in the process. Greensgrow, a one-acre plot of raised beds and greenhouses on the site of a former steel-galvanizing factory, is turning a profit by selling its own vegetables and herbs as well as a range of produce from local growers, and by running a nursery selling plants and seedlings. The farm earned about $10,000 on revenue of $450,000 in 2007, and hopes to make a profit of 5 percent on $650,000 in revenue in this, its 10th year, so it can open another operation elsewhere in Philadelphia.*

We defined precision and recall measures on the LF. Given a gold-standard LF-graph, we can evaluate the LF graph produced by a system by defining node and edge scoring criteria and then computing the node alignment that maximizes the overall score. The evaluation metric between a gold LF graph G and a test LF graph T is then defined as the maximum score produced by any node/edge alignment from the gold to the test LF.

We parsed the seven texts to obtain the LF-graphs for each. Then we took each paragraph and hand-built a gold-standard LF-graph for each. Using the precision and recall measures discussed above, the base parser attained 61.4% precision and 67.2% recall on the unseen data.

	Initial Baseline System	*Baseline System after devel*	*w/ NER, POS, and UKW lookup*	*w/ constituent advice from Stanford Parser*
Prec.	61.40%	74.4%	78.2%	79.0%
Recall	67.20%	74.4%	82.6%	82.8%

Table 1: Evaluation on combined texts

We then performed a limited amount of lexical and grammatical development based on the evaluation: adding 26 new lexical items (17 nouns, 1 verb, 7 adjectives and 1 adverb), 33 new or modified senses for existing lexical items, 7 new ontology concepts, and two grammar rules. Word sense modifications included adding a new argument structure pattern to a lexical entry and/or a new semantic role to an existing concept. For example, in some cases an ontology concept included an agent role, but not one for a more general cause role. We did not attempt to add all unknown words and senses; Aside from the proper nouns, there are still 14 unknown words (e.g., *merchandising*, *propellant*, *nitrocellulose*) remaining in the texts for which we derive entries for analysis from unknown word lookup.

After this development, we estimated the potential of the extended system by rerunning the base parser and then several combinations of the extensions. By adding named-entity recognition, unknown word lookup, and part of speech tagging advice, performance rises to 78.2% precision and 82.6% recall. Adding advice on constituent bracketing using the Stanford parser gave only a slight improvement. These results are summarized in Table 1. Because almost all prior work has not attempted an evaluation of deep understanding, there is little prior work to compare to. However, just on the face of the scores, we think we have made a convincing case that domain-independent, broad-coverage, deep understanding of language is a technology within reach.

3. Using A Collaborative Problem Solving Agent for One-shot Task Learning

We developed the generic collaborative problem solving model using several different applications as test cases. The most significant system is one that focuses on agents that can acquire the task models they need from intuitive language-rich demonstrations by humans. These agents use the same collaborative architecture to learn tasks as they do to perform tasks. The system displays an integrated intelligence that results from sophisticated natural language understanding, reasoning, learning, and acting capabilities unified within a collaborative agent architecture.

Background on Task Learning

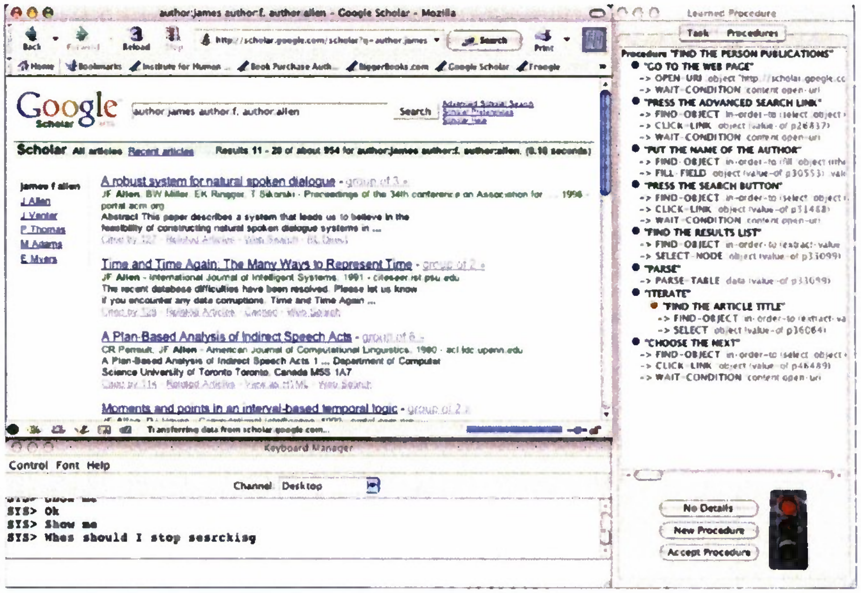
In previous work, researchers have attempted to learn new tasks by observation, creating agents that learn through observing an expert's demonstration (Angros et al. 2002, Lau & Weld 1999; Lent & Laird 2001). These techniques require observing multiple examples of the same task, and the number of training examples required increases dramatically with the complexity of the task. To be effective, however, collaborative assistants need to be able to acquire tasks much more quickly – typically from a single example, possibly with some clarification dialogue. To enable this, in our system the teacher not only demonstrates the task, but also gives a "play-by-play" description of what they are doing. This is a natural method that people already use when teaching other people, and our system exploits this natural capability. By combining the information from understanding with prior knowledge and a concrete example demonstrated by the user, our system (called PLOW) can learn complex tasks involving iterative loops in a single short training session.

PLOW learns tasks that can be performed within a web browser. These are typically information management tasks, e.g., finding appropriate sources, retrieving information, filing requisitions, booking flights, and purchasing things, Figure 3 shows the user interface as it was used in the evaluation. The main window on the left is simply the Mozilla browser, instrumented so that PLOW can monitor user actions. On the right is the procedure that PLOW has learned so far, summarized back in language from the task model using PLOW's language generation capabilities. Across the bottom is a chat window that shows the most recent interactions. The user can switch between speech and keyboard throughout the interaction.

The Agent Architecture

The understanding components combine natural language (speech or keyboard) with the observed user actions on the GUI. After full parsing, semantic interpretation and discourse interpretation produce plausible intended actions. These are passed to the collaborative problem solving (CPS) agent, which settles on the most likely intended interpretation given then current problem solving context. Depending on the actions, the CPS agent then drives other parts of the system. For example, if the recognized user action was to demonstrate the next step in the task, the CPS agent invokes the task learning, which if successful will update the task models in the knowledge base. If, on the other hand, the recognized user intent was to request the execution of a (sub)task, the CPS agent attempts to look up a task that can accomplish this action in the knowledge base. It then invokes the execution system to perform the task. During collaborative learning, the system may actually do both – it may learn a new step in the task being learned, but because it already knows how to do the subtask, it performs it for the user. This type of collaborative execution while learning is critical in enabling the learning of iterative steps without requiring the user to tediously demonstrate each loop through the iteration.

While we have shown examples of how integrating language, dialogue, reasoning and learning has great potential for effective one-shot task learning, the real test is whether ordinary users can quickly learn to use the system to teach new procedures. There are many possible pitfalls: (1) do we have comprehensive enough natural language understanding capabilities so that users expressing information in intuitive ways are likely to be understood? (2) can we really learn robust task models from a single example, (3) can the users easily determine whether the system is learning correctly as they are


author:james author:f. author:allen - Google Scholar - Mozilla
Back
Forward
Reload
Stop
http://scholar.google.com/scholar?q=author:james
Search
Print
Google Scholar
author:james author:f. author:allen
Search
Scholar All articles Recent articles Results 11 - 20 of about 954 for author:james author:f. author:allen. (0.10 seconds)
A robust system for natural spoken dialogue
Time and Time Again: The Many Ways to Represent Time
A Plan-Based Analysis of Indirect Speech Acts
Moments and points in an interval-based temporal logic
Transferring data from scholar.google.com...
Learned Procedure
Task
Procedures
Keyboard Manager
Control Font Help
Channel: Desktop
No Details
New Procedure
Accept Procedure

```
Procedure "FIND THE PERSON PUBLICATIONS"
● "GO TO THE WEB PAGE"
  -> OPEN-URI object "http://scholar.google.cc
  -> WAIT-CONDITION content open-uri
● "PRESS THE ADVANCED SEARCH LINK"
  -> FIND-OBJECT in-order-to (select object
  -> CLICK-LINK object (value-of p26837)
  -> WAIT-CONDITION content open-uri
● "PUT THE NAME OF THE AUTHOR"
  -> FIND-OBJECT in-order-to (fill object (the
  -> FILL-FIELD object (value-of p30553) val
● "PRESS THE SEARCH BUTTON"
  -> FIND-OBJECT in-order-to (select object
  -> CLICK-LINK object (value-of p31488)
  -> WAIT-CONDITION content open-uri
● "FIND THE RESULTS LIST"
  -> FIND-OBJECT in-order-to (extract value
  -> SELECT-NODE object (value-of p33099)
● "PARSE"
  -> PARSE-TABLE data (value-of p33099)
● "ITERATE"
    ● "FIND THE ARTICLE TITLE"
      -> FIND-OBJECT in-order-to (extract va
      -> SELECT object (value-of p36064)
● "CHOOSE THE NEXT"
  -> FIND-OBJECT in-order-to (select object
  -> CLICK-LINK object (value-of p46489)
  -> WAIT-CONDITION content open-uri
```

```
SYS> Ok
SYS> Show me
SYS> Whes should I stop searching
```

Figure 3: The PLOW Interface

teaching the system.

In August 2006, we delivered a version of the PLOW system to independently contracted evaluators. At that point, we had developed the system to ensure that we (the developers) could effectively teach PLOW to learn how to answer seventeen pre-determined test question templates. The evaluators recruited 16 test subjects who received general training on how to use PLOW and many other applications that were part of the overall project evaluation. Among these were three other task learning systems: one learns entirely from passive observation, one used a sophisticated GUI primarily designed for editing procedures but extended to allow the definition of new procedures, and the third used an NL-like query and specification language that required users to have a detailed knowledge of HTML producing the web pages.

After training, the subjects then performed the first part of the test, in which they had to use different systems to teach some subset of the predefined test questions. Seven of these involved the PLOW system. Once the procedures were learned by the systems, the evaluators created a set of new test examples by specifying values for the input parameters to the task and then scored the results from executing the learned task models using predetermined scoring metrics individualized to each question. The PLOW system did well on this test, scoring 2.82 out of 4 across all test questions and the 16 subjects.

The second part of the test involved a set of 10 new "surprise" test questions not previously seen by any of the developers (see Figure 1). Some of these were close variants to the original test questions, and some were entirely new tasks. The sixteen subjects had one work day to teach whichever of these surprise tasks they wished, using whichever of the task learning systems they wished. As a result, this test reveals not only the core capability for learning new tasks, but also evaluates the usability of the four task learning systems.

PLOW did very well on this test on all measures. Out of the 16 users, thirteen of them used PLOW to teach at least one question. Of the other systems, the next most used system was used by eight users. If we look at the total number of tasks successfully taught, we see that PLOW was used to teach 30 out of the 55 task models that were constructed during the day. Furthermore, the tasks constructed using PLOW received the highest average score in the testing (2.2 out of 4).

Concluding Remarks

This project has developed significant generic technology for dialogue systems that is reusable across domains. We have developed and demonstrated the potential of broad-covergae deep language understanding, and developed a generic collaborative problems solving architecture that can enable sophisticated mixed-intiative dialogue,. As demonstrated in the PLOW system.

References

Allen, J., M. Swift, et al. (2008). Deep Semantic Analysis of Text. Symposium on Semantics in Systems for Text Processing (STEP). Venice, Italy.

Allen, J. F., N. Chambers, et al. (2007). PLOW: A collaborative task learning agent. *Named Best Paper,* National Conference on Artificial Intelligence (AAAI). Vancouver, BC.

Allen, J. F., M. Dzikovska, et al. (2007). Deep linguistic processing for spoken dialogue systems. Workshop on Deep Linguistic Processing, Association for Computational Linguistics. Prague.

Angros, R.; Johnson, L.; Rickel, J.; and Scholer A. 2002. Learning Domain Knowledge for Teaching Procedural Skills, *Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems.*

Blythe, J. 2005. Task Learning by Instruction in Tailor. *Proceedings of the Joint Conference on Autonomous Agents and Multiagent Systems.*

Blythe, J. 2005. Task Learning by Instruction in Tailor. *Proceedings of the International Conference on Intelligent User Interfaces.*

Johan Bos and Rodolfo Delmonte (Eds.), Semantics in Text Processing. *STEP 2008 Conference Proceedings, vol. 1 of Research in Computational Semantics*: 277–286. College Publications.

Chambers, N.; Allen, J.: Galescu, L.; Jung, H.; and Taysom, W. 2006. Using Semantics to Identify Web Objects. *Proceedings of the National Conference on Artificial Intelligence.*

Dzikovska, M., J. F. Allen, et al. (2008). "Linking Semantic and Knowledge Representation in a Multi-Domain Dialogue System." Logic and Computation **18**(3): 405-430.

Dzikovska, M., M. Swift, et al. (2007). Customizing meaning: Building domain-specific semantic representations from a generic lexicon. Computing Meaning. H. Bunt and R. Muskens, Springer. **3:** 213-231.

Jung, H., J. F. Allen, et al. (2008). "Utilizing natural language for one-shot task learning." Logic and Computation **18**(3): 475-493.

Jung, H., J. Allen, et al. (2006). One-Shot Procedure Learning from Instruction and Observation. FLAIRS, Melbourne, FL.

Lau, T.; Bergman, L.; Castelli, V.; and Oblinger, D. 2004 Sheep Dog: Learning Procedures for Technical Support, *Proceedings of the International Conference on Intelligent User Interface.*

Lau, T. and Weld, D. 1999. Programming by Demonstration: An Inductive Learning Formulation. *Proceedings of the International Conference on Intelligent User Interfaces.*

Lee F.; and Anderson, J. 1997 Learning to act: Acquisition and Optimization of Procedural Skill, *Proceedings of the Annual Conference of the Cognitive Science Society.*

Lent, M. and Laird, J. 2001. Learning Procedural Knowledge through Observation, *Proceedings of the International Conference on Knowledge Capture.*

Ann Copestake, Dan Flickinger, Carl Pollard and Ivan Sag. 2005. Minimal Recursion Semantics: An Introduction. *Research on Language and Computation*, 3(4):281-332.

Donald Davidson. 1967. The logical form of action sentences. In Nicholas Rescher, (Ed.), *The Logic of Decision and Action,* pp. 81–95. U. of Pittsburg Press.

Christopher Johnson and Charles Fillmore. 2000. The FrameNet tagset for frame-semantic and syntactic coding of predicate-argument structure. *ANLP-NAACL*, Seattle, WA.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlcxicalized Parsing. *ACL* :423-430.

Catherine Macleod, Ralph Grishman, and Adam Meyers. 1994. Creating a common syntactic dictionary of English. *International Workshop on Sharable Natural Language Resources*, Nara.

Dan Melamed and Philip Resnik 2000. Tagger Evaluation Given Hierarchical Tag Sets, *Computers and the Humanities* 34(1-2).

George Miller. 1995. Wordnet: A lexical database for English. *Comm. ACM* 38(5).

Philip Resnik and David Yarowsky. 1997. A Perspective on Word Sense Disambiguation Methods and their Evaluation. *SIGLEX Workshop on Tagging Text, ANLP-97.*

Mary Swift, James Allen, and Daniel Gildea. 2004. Skeletons in the parser: Using a shallow parser to improve deep parsing. *COLING*, Geneva.